

# Private Learning on Networks: Part II \*

Shripad Gade      Nitin H. Vaidya

Department of Electrical and Computer Engineering, and  
Coordinated Science Laboratory,  
University of Illinois at Urbana-Champaign.  
Email: {gade3, nhv}@illinois.edu

Technical Report

## Abstract

Widespread deployment of *distributed* machine learning algorithms has raised new privacy challenges. The focus of this paper is on improving privacy of each participant's local information (such as dataset or loss function) while collaboratively learning underlying model. We present two iterative algorithms for privacy preserving distributed learning. Our algorithms involves adding structured randomization to the state estimates. We prove deterministic correctness (in every execution) of our algorithm despite the iterates being perturbed by non-zero mean random variables. We motivate privacy using privacy analysis of a special case of our algorithm referred to as Function Sharing strategy (presented in [1]).

## 1 Introduction

Advances in fast learning algorithms and efficient computing hardware have resulted in widespread deployment of machine learning systems. Distributed learning and inference have become popular due to their inherent efficiency, scalability, robustness and geo-distributed nature of datasets [2–6]. In a distributed machine learning scenario, partitions of the dataset are stored among several different agents (such as servers or mobile devices, [7]), and these agents communicate and collaboratively solve a distributed optimization problem in order to collectively learn the most appropriate “model parameters”. A suitable cost function is specified for this purpose, and the most appropriate *model parameters* are the ones that minimize this cost function. Distributed learning using distributed optimization can reduce communication requirements of learning, since the agents need to communicate information (such as gradients or estimates) that are often much smaller in size than the datasets.

Many distributed optimization algorithms have appeared in the literature over the past decade [8–12]. Solutions to distributed optimization of convex functions have been proposed for myriad scenarios involving directed graphs [13], communication link failures and losses [14], asynchronous communication models [15–17], stochastic objective functions [18, 19], and fault tolerance [20].

The distributed optimization algorithms rely on exchange of information between the different agents. However, this makes the distributed algorithms vulnerable to privacy violations. In particular, information about an agent's local dataset may become known to other agents through the information exchanged between the agents. Such privacy concerns have emerged as a critical challenge in machine learning [21–25]. For instance, in the healthcare industry, hospitals and insurance providers use medical records to learn predictive models estimating individual risk towards certain diseases. Personalized recommendation systems is another application in which privacy has emerged as a concern. For instance, recommendation systems for movies or products, and applications that suggest the next word a user will type (e.g., SwiftKey<sup>1</sup>) make use

---

\*This research is supported in part by National Science Foundation awards 1421918 and 1610543, and Toyota InfoTechnology Center. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government. <sup>1</sup> <https://swiftkey.com/en>

of user data for training models. While such machine learning applications can improve user experience, they also threaten the user’s privacy [26]. Motivated by these examples in the context of distributed optimization, we ask the following question:

Can agents collaboratively learn underlying model parameters without leaking private information?

The paper presents two privacy-preserving algorithms for distributed optimization, which can be used for improving privacy in distributed machine learning. Although our work is motivated by machine learning applications, the proposed solutions have applications wherever distributed optimization formulation is adopted.

We consider a distributed optimization problem involving  $S$  agents, each of whom has access to a local convex objective function  $f_i(x)$ . In the context of machine learning for classification, the local objective function may be a *loss function* that measures the accuracy of classification on the training dataset using a given choice of *model parameters* – here  $x$  denotes the vector of model parameters. As an example, in the context of classification task,  $f_i(x)$  may denote the *logistic loss function* for the data items stored at agent  $i$  – the loss depends on the parameters of the classification hypothesis, and the goal is to identify parameters that minimize the loss over all the agents (i.e., over data stored at all the agents). In Section 6.2, we will elaborate on the application of our work to machine learning.

**Problem 1.** *The set of  $S$  agents need to distributedly compute the optimum of the global objective function, which consists of the sum of the local objective functions. That is,*

$$\text{Distributedly find } x^* \in \underset{x \in \mathcal{X}}{\operatorname{argmin}} \sum_{i=1}^S f_i(x) \quad (1)$$

where  $\mathcal{X}$  is the set that contains all feasible values for parameter vector  $x$ .

The local objective function  $f_i(x)$  at each agent  $i$  is assumed to be a convex function. Additional assumptions regarding the objective functions and the communication network interconnecting the agents are detailed later.

## 1.1 Contributions

In this paper we present three algorithms for privacy-preserving distributed optimization:

- Randomized State Sharing (RSS, Algorithm 1) :
  - Our privacy-preserving algorithm uses *randomization*. However, unlike differential privacy schemes, our strategy preserves optimality by introducing correlation between the randomness added to local model parameter estimates.
  - We prove asymptotic convergence in a deterministic setting (every execution) and argue its privacy using the privacy analysis developed for a special case.
- Function Sharing (FS, Algorithm 4):
  - We show that Function Sharing strategy (Algorithm 4, presented in [1]) simulates a special case of RSS algorithm. If the random perturbations added to local iterates in RSS algorithm are state dependent, then the RSS algorithm imitates FS algorithm.
  - The deterministic convergence is shown to easily follow from the convergence analysis for RSS Algorithm.
- Randomized State Sharing - Locally Balanced (RSS-LB, Algorithm 3):
  - RSS-LB is a distributed learning algorithm that uses locally balanced randomization to perturb the parameter estimates (perturbations add to zero at each node). Unlike RSS, agents do not share the same perturbed estimate with neighbors. Neighbors receive dissimilar estimates from agent  $j$ .
  - We show deterministic convergence of RSS-LB.

## 1.2 Related Work

Privacy-preserving methods for machine learning can be broadly classified into cryptographic approaches and non-cryptographic approaches [27]. Cryptography-based privacy preserving optimization algorithms [28–32] tend to be computationally expensive.

Non-cryptographic approaches have gained popularity in recent years.  $\epsilon$ -*differential privacy* is a probabilistic technique that involves use of randomized perturbations [22, 33–36] to minimize the probability of uncovering of specific records from statistical databases. Differential privacy methods, however, suffer from a fundamental trade-off between the accuracy of the solution and the privacy margin (parameter  $\epsilon$ ). *Transformation* is another non-cryptographic technique that involves converting a given optimization problem into a new problem via algebraic transformations such that the solution of the new problem is the same as the solution of the old problem. This enables agents to conceal private problem data (of the original problem) effectively while the quality of solution is preserved [27, 37]. Scaling, translation, affine transformation and certain nonlinear transformations have been suggested in the literature [27, 37–40]. Transformation approaches in literature, however, cater to specific problems.

Some researchers [41] have claimed that asynchronous algorithms can be used to improve privacy, due to timing unpredictability of agent updates. However, they do not provide a formal proof of privacy improvement. To the extent that asynchrony by itself can improve privacy, our solutions below can also be extended to asynchronous settings to obtain similar benefits.

Authors, in [42], use Lyapunov analysis to show strong convexity of  $f(x)$  and  $N$ -Lipschitzness of  $\nabla f_i(x)$  is sufficient<sup>2</sup> for convergence of distributed gradient descent algorithm. Our analysis provides an alternate proof for weaker assumption on  $f(x)$ <sup>3</sup>, albeit with additional requirement of bounded gradients (cf. [43]). In our prior report, [44], we show that by multiplying gradients with structured random weights, clients (workers) can hide their gradients in a multi-parameter server architecture. In a companion report [1], we propose a function sharing strategy (FS) that involves using noise functions to obfuscate true objective functions  $f_i(x)$ . We show convergence of the iterate to the optimum (correctness) and further prove that under a  $f+1$  vertex connectivity of underlying communication topology, the learning system is private to an adversary that can corrupt at most  $f$  nodes.

**Organization:** The rest of the paper is organized as follows. Problem formulation is presented in Sec. 2 followed by our privacy-preserving distributed learning (RSS and RSS-LB) algorithms in Sec. 3. Convergence analysis is presented in Sec. 4. Our third privacy preserving learning (FS) algorithm is presented in Sec. 5. Discussion on privacy analysis and applications to machine learning are presented in Sec. 6

## 2 Problem Formulation

### 2.1 Notation

We consider a *synchronous* system consisting of  $S$  agents (also referred to as nodes) connected in a network of bidirectional communication links. The communication links are always reliable. The set of agents is denoted by  $\mathcal{V}$ ; thus,  $|\mathcal{V}| = S$ . For convenience, we view each bidirectional link as consisting of two directed links. Thus, if agents  $u$  and  $v$  are connected, there is a directed link from  $u$  to  $v$  denoted as  $u \rightarrow v$ , and a directed link from  $v$  to  $u$  denoted as  $v \rightarrow u$ . Define  $\mathcal{E}$  as a set of directed edges corresponding to the directed communication links:  $\mathcal{E} = \{(u, v) : u \in \mathcal{V}, v \in \mathcal{V} \text{ and } u \rightarrow v\}$ . Communication links between the agents induce a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , defined by the set of all nodes (agents),  $\mathcal{V}$  and the set of directed edges (communication links),  $\mathcal{E}$ . The neighborhood set of agent  $v$  is defined as the set of agents that are connected to agent  $v$ . By convention,  $\mathcal{N}_v$  includes  $v$  itself, i.e.  $\mathcal{N}_v = \{u \mid (u, v) \in \mathcal{E}\} \cup \{v\}$ .  $v$  can both send and receive messages from agents in  $\mathcal{N}_v$  (bidirectional links).

Let  $D$  denote the dimension of the optimization problem (i.e. dimension of vector  $x$ ).  $\|\cdot\|$  denotes Euclidean norm for vectors. Let  $\mathcal{X}^*$  denote the set of all optima of  $f(x)$ , and let  $f^*$  denote the optimal value of  $f(x)$  over  $x \in \mathcal{X}$ .

$$f^* = \inf_{x \in \mathcal{X}} f(x) \text{ and } \mathcal{X}^* = \{x \in \mathcal{X} \mid f(x) = f^*\}. \quad (2)$$

---

<sup>2</sup>  $f_i(x)$  need not be convex. <sup>3</sup>  $f(x)$  only needs to be convex.

Iteration number is denoted by  $k$ . The model parameter vector (iterate) stored with agent  $i$  at time (iteration)  $k$  is denoted by  $x_k^i$ , where the superscript denotes the agent-id, the subscript denotes the time index. The iterate average at time instant  $k$  is denoted by  $\bar{x}_k$  and the disagreement of an iterate at agent  $j$  ( $x_k^j$ ) with the iterate average ( $\bar{x}_k$ ) by  $\delta_k^j$ .

$$\bar{x}_k = \frac{1}{S} \sum_{j=1}^S x_k^j, \quad \delta_k^j = x_k^j - \bar{x}_k. \quad (3)$$

## 2.2 Problem Setup

Consider a distributed learning problem as defined in Problem 1, involving  $S$  agents collaboratively learning the optimizer of  $f(x)$ . Each of the  $S$  agents has access to a application specific objective function  $f_i(x)$ .  $x \in \mathcal{X}(\subseteq \mathbb{R}^D)$  denotes the parameter vector, and  $\mathcal{X}$  represents the set of all feasible parameter vectors. We make the following assumptions on the functions  $f_i(x)$  and on the feasible parameter set  $\mathcal{X}$ .

**Assumption 1** (Objective Function and Feasible Set).

- (A) The objective function  $f_i : \mathbb{R}^D \rightarrow \mathbb{R}$ , is a convex function of model parameter vector  $x$  ( $\forall i$ ). It follows that  $f(x) := \sum_{i=1}^S f_i(x)$  is also a convex function.
- (B) The feasible parameter vector set,  $\mathcal{X}$ , is a non-empty, convex, and compact subset of  $\mathbb{R}^D$ .

For  $z \in \mathbb{R}^D$ , we define projection operator  $\mathcal{P}_{\mathcal{X}}$  as follows,  $\mathcal{P}_{\mathcal{X}}(z) = \arg \min_{y \in \mathcal{X}} \|z - y\|$ . We further make assumptions regarding the gradient of individual function  $f_i(x)$ .

**Assumption 2** (Gradient Bound and Lipschitzness).

- (A) The gradients are norm bounded, i.e.  $\exists$  scalar  $L > 0$  such that,  $\|\nabla f_h(x)\| \leq L$ ,  $\forall x \in \mathcal{X}$  and  $\forall h$ .
- (B) The gradients are Lipschitz continuous i.e.  $\exists$  scalar  $N > 0$  such that,  $\|\nabla f_h(x) - \nabla f_h(y)\| \leq N\|x - y\|$ , for all  $x \neq y$  ( $x, y \in \mathcal{X}$ ) and  $\forall h$ .

We assume the communication graph  $\mathcal{G}$  to be connected. We may need to impose additional connectivity constraint for analyzing privacy (e.g.  $f$ -admissibility condition from Theorem 3 in [1]).

**Assumption 3.**  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  is a connected graph.

## 3 Algorithm

In this section, we first present iterative distributed learning algorithm (DGD) from [10]. Later we show how we modify DGD to make it privacy-preserving.

### 3.1 Algorithm 0 - DGD

Recently popularized iterative distributed descent algorithms (e.g. Distributed Gradient Descent DGD) use a combination of consensus dynamics and local gradient descent to find a minimizer of  $f(x)$ . More precisely, in every iteration, agents receive parameter estimates from neighbors and perform consensus step followed by descent along local gradient direction (cf. Algorithm 0). Such algorithms are not privacy-preserving by design and adversaries may uncover private data by observing information exchange.

The *consensus step* (also called *information fusion*), involving convex combination of received state estimates, is a weighted averaging step (Line 5, Algorithm 0). The weights  $B_k[j, i]$  denotes the weight assigned by agent ' $j$ ' to the estimate received from agent ' $i$ '. If  $j$  does not receive estimate from  $i$ , i.e.  $i \notin \mathcal{N}_j$ , then  $B_k[j, i] = 0$ . The weight matrix  $B_k$  can be selected by agents using established methods like Metropolis-Hastings Algorithm [45, 46], Equal Neighbor Weights protocol [47] or protocol in [48] (all edges have equal weight). The smallest non-zero entry in  $B_k$  matrix (for any  $k$ ) is denoted by  $\rho$ .

The *projected gradient descent* step (Line 6, Algorithm 0) involves descent along local gradient ( $\nabla f_j(v_k^j)$ ) followed by projection onto feasible set. The learning step-sizes (denoted as  $\alpha_k$  at iteration  $k$ ) form a positive

---

**Algorithm 0** DGD Algorithm [10]

---

```
1: For each agent  $j$  in parallel:
2: for  $k = 1$  to MAX_ITER do
3:   Send estimate  $x_k^j$  to agents  $i \in \mathcal{N}_j$ 
4:   Receive estimate  $x_k^i$  from agents  $i \in \mathcal{N}_j$ 
5:   Information Fusion:
        $v_k^j = \sum_{i \in \mathcal{N}_j} B_k[j, i] x_k^i$ 
6:   Projected Gradient Descent:
        $x_{k+1}^j = \mathcal{P}_{\mathcal{X}} \left[ v_k^j - \alpha_k \nabla f_j(v_k^j) \right]$ 
7: end for
```

---

( $\alpha_k > 0$ ), non-increasing ( $\alpha_{k+1} \leq \alpha_k, \forall k \geq 0$ ), deterministic sequence (function of  $k$ ) that is non-summable, yet square summable.

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \text{ and } \sum_{k=0}^{\infty} \alpha_k^2 < \infty. \quad (4)$$

The local gradient descent behaves like perturbations to consensus dynamics over the state estimates and hence its effect gets averaged. This averaging ensures that the iterates eventually move in the direction of  $-\nabla f(x)$  and asymptotic convergence of the iterates to the optimum of  $f(x)$ .

### 3.2 RSS Algorithm

The Randomized State Sharing (RSS, Algorithm 1) algorithm is a modified version of Algorithm 0, in that each agent shares a perturbed version of its estimate to neighbors during consensus step, as discussed below.

Initially, each agent  $j$  generates random number  $d_k^j$  (Line 3, Algorithm 1) such that  $\sum_{j=1}^S d_k^j = 0$  and  $|d_k^j| \leq \Delta$ , where  $\Delta$  is a constant bound. We present a simple strategy to generate such correlated random numbers ( $d_k^j$ ) as Algorithm 2. Each agent  $j$  then computes a perturbed version of its estimate (Line 4, Algorithm 1).

$$w_k^j = x_k^j + \alpha_k d_k^j \quad (5)$$

The perturbation ( $\alpha_k d_k^j$ ) is obtained by multiplying the random number  $d_k^j$  and the learning step-size  $\alpha_k$  (at iteration  $k$ ).  $\alpha_k$  has same properties as defined in Algorithm 0.

Next, each agent  $j$  send ( $w_k^j$ ) and receive ( $w_k^i$ ) obfuscated estimates from neighbors ( $i \in \mathcal{N}_j$ ) (Line 5-6, Algorithm 1). Each agent  $j$  then performs *information fusion*, involving convex combination of received state estimates, as codified in doubly stochastic matrix  $B_k$  as follows,

$$v_k^j = \sum_{i \in \mathcal{N}_j} B_k[j, i] w_k^i. \quad (6)$$

The information fusion step is followed by *projected gradient descent*, involving descent along local gradient, and it is formally written as,

$$x_{k+1}^j = \mathcal{P}_{\mathcal{X}} \left[ v_k^j - \alpha_k \nabla f_j(v_k^j) \right]. \quad (7)$$

The learning rate  $\alpha_k$  has same definition as in Algorithm 0.

#### Generating $d_k^j$

Inspired by Secure Multi-party Computation (SMC) strategies [23], we present a decentralized protocol (Algorithm 2) to compute correlated random perturbations  $d_k^j$ . In each round, every agent  $j$  picks a random number from an arbitrary probability distribution, such that  $|s_k^{j,i}| \leq \frac{\Delta}{2S}$  where  $\Delta$  is a constant bound and

---

**Algorithm 1** Randomized State Sharing (RSS) Algorithm

---

- 1: For each agent  $j$  in parallel:
  - 2: **for**  $k = 1$  to MAX\_ITER **do**
  - 3:   Agent  $j$  chooses a random number  $d_k^j$  such that,  
         $\sum_{j=1}^S d_k^j = 0, \quad \forall k$
  - 4:   Agent  $j$  computes perturbed state estimate  $w_k^j$ :  
         $w_k^j = x_k^j + \alpha_k d_k^j$
  - 5:   Send  $w_k^j$  to each neighbor  $i \in \mathcal{N}_j$ .
  - 6:   Receive  $w_k^i$  from each  $i \in \mathcal{N}_j$ .
  - 7:   Information Fusion:  
         $v_k^j = \sum_{i \in \mathcal{N}_j} B_k[j, i] w_k^i$
  - 8:   Projected Gradient Descent:  
         $x_{k+1}^j = \mathcal{P}_{\mathcal{X}} \left[ v_k^j - \alpha_k \nabla f_j(v_k^j) \right]$
  - 9: **end for**
- 

---

**Algorithm 2** SMC Algorithm for Generating Correlated Perturbation ( $d_k^j$ ), [Line 3, Algorithm 1]

---

- 1: For each agent  $j$  in parallel (at each  $k$ ):
  - 2: Agent  $j$  transmits  $s_k^{j,i}$  to neighbor  $i \in \mathcal{N}_j$
  - 3: Agent  $j$  receives  $s_k^{i,j}$  from neighbor  $i$  ( $j \in \mathcal{N}_i$ )
  - 4: Construct  $d_k^j$ :  $d_k^j = \sum_{i: j \in \mathcal{N}_i} s_k^{i,j} - \sum_{i \in \mathcal{N}_j} s_k^{j,i}$
- 

$S$  is the total number of agents. Next, every agent  $j$  shares random number  $s_k^{j,i}$  with each of its neighbors  $i$  ( $i \in \mathcal{N}_j$ ), and receives random number  $s_k^{i,j}$ . The sharing of  $s_k^{i,j}$  is secure by assumption. Next, every agent  $j$  generates the correlated random number ( $d_k^j$ ) by adding all the random numbers that it has received from neighbors ( $s_k^{i,j}$ ) and subtracting all the random numbers that it has shared with its neighbors ( $s_k^{j,i}$ ). As shown in Line 4 of Algorithm 2,

$$d_k^j = \sum_{i \in \mathcal{N}_j} s_k^{i,j} - \sum_{i \in \mathcal{N}_j} s_k^{j,i}. \quad (8)$$

It is easy to see that the sum of all correlated random numbers is zero, i.e.  $\sum_j d_k^j = 0$  ( $\forall k$ ).

Since  $s_k^{i,j}$  are bounded ( $\forall i, j$  at all  $k$ ), the correlated random perturbation  $d_k^j$  above, is bounded as  $|d_k^j| \leq \Delta$ .

### 3.3 RSS-LB Algorithm

The Randomized State Sharing - Locally Balanced algorithm (RSS-LB) is a distributed algorithm that aims to improve privacy by sharing differently perturbed parameter estimates with neighbors.

Initially, each agent generates random number  $d_k^{j,i}$  (Line 3, Algorithm 3) such that  $\sum_{i \in \mathcal{N}_j} d_k^{j,i} = 0$  for all  $k$ . Each random number is bounded by a constant number, i.e.  $|d_k^{j,i}| \leq \Delta$ . Since the random numbers generated by every agent add up to zero, generating  $d_k^{j,i}$  does not require any sophisticated protocol (unlike RSS-GB algorithm). Agent  $j$  then computes several perturbed versions of its estimate to be shared with neighbors (Line 4, Algorithm 3).

$$w_k^{j,i} = x_k^j + \alpha_k d_k^{j,i}. \quad (9)$$

The perturbation ( $\alpha_k d_k^{j,i}$ ) is obtained by multiplying the random number  $d_k^{j,i}$  and the learning step-size  $\alpha_k$  (at iteration  $k$ ).  $\alpha_k$  has same properties as defined in Algorithm 0. Note that each neighbor receives a different estimate from agent  $j$ . Per definition,  $w_k^{j,j} = x_k^j$ . This implies that agents do not add noise to their own estimate during algorithm execution. The noise is used to only perturb the shared states. Moreover,

---

**Algorithm 3** Randomized State Sharing (RSS-LB) Algorithm

---

- 1: For each agent  $j$  in parallel:
  - 2: **for**  $k = 1$  to MAX\_ITER **do**
  - 3:   Agent  $j$  chooses a random number  $d_k^{j,i}$  such that,  
    
$$\sum_{i \in \mathcal{N}_j} d_k^{j,i} = 0, \quad \forall k$$
  - 4:   Agent  $j$  computes perturbed state estimate  $w_k^{j,i}$ :  
    
$$w_k^{j,i} = x_k^j + \alpha_k d_k^{j,i}$$
  - 5:   Send  $w_k^{j,i}$  to each neighbor  $i \in \mathcal{N}_j$ .
  - 6:   Receive  $w_k^{i,j}$  from each neighbor  $i \in \mathcal{N}_j$ .
  - 7:   Information Fusion:  
    
$$v_k^j = \sum_{i \in \mathcal{N}_j} B_k[j, i] w_k^{i,j}$$
  - 8:   Projected Gradient Descent:  
    
$$x_{k+1}^j = \mathcal{P}_{\mathcal{X}} \left[ v_k^j - \alpha_k \nabla f_j(v_k^j) \right]$$
  - 9: **end for**
- 

we assume that the noise  $(d_k^{i,j})$  generated by any agent  $i$  satisfies  $\sum_{j=1}^S B_k[j, i] d_k^{i,j} = 0$ .<sup>4</sup> This assumption implies  $\sum_{i=1}^S (\sum_{j=1}^S B_k[j, i] d_k^{i,j}) = 0$ , that is  $\sum_{j=1}^S \sum_{i=1}^S B_k[j, i] d_k^{i,j} = \sum_{j=1}^S e_k^j = 0$ .

**Assumption 4.** The perturbation  $(d_k^{i,j})$  generated by agent ‘ $i$ ’ is assumed to satisfy  $\sum_{j=1}^S B_k[j, i] d_k^{i,j} = 0$  for all  $k$ .

Next, each agent  $j$  sends  $(w_k^{j,i})$  and receives  $(w_k^{i,j})$  obfuscated estimates from neighbors ( $i \in \mathcal{N}_j$ ) (Line 5-6, Algorithm 3). Each agent  $j$  then performs *information fusion*, involving convex combination of received state estimates, as codified in doubly stochastic matrix  $B_k$ , as follows,

$$v_k^j = \sum_{i \in \mathcal{N}_j} B_k[j, i] w_k^{i,j}. \quad (10)$$

The information fusion step is followed by *projected gradient descent*, involving descent along local gradient, and it is formally written as,

$$x_{k+1}^j = \mathcal{P}_{\mathcal{X}} \left[ v_k^j - \alpha_k \nabla f_j(v_k^j) \right]. \quad (11)$$

The learning rate  $\alpha_k$  has same definition as in Algorithm 0.

### Differences between RSS and RSS-LB

- In RSS-LB, agents share dissimilar perturbed estimates with neighbors. On the contrary RSS algorithm involves sharing same perturbed estimate with all neighbors.
- In RSS-LB, agents do not add perturbations to their own parameter estimates. Perturbed state estimates are only used to share information with neighbors.

## 4 Analysis

### 4.1 RSS Convergence Analysis

The RSS algorithm (Algorithm 1) involves perturbed information fusion and projected gradient descent operations. The update equation (Line 7, Algorithm 1) can be rewritten as the sum of fusion of true states,  $\hat{v}_k^j$ , and the fusion of random perturbations,  $e_k^j$ , as follows:

$$v_k^j = \sum_{i \in \mathcal{N}_j} B_k[j, i] w_k^i = \hat{v}_k^j + \alpha_k e_k^j, \quad (12)$$

---

<sup>4</sup> In fact if  $B_k$  matrix has special structure (as discussed in Section 4.2) then any locally balanced perturbation will work.

$$\hat{v}_k^j = \sum_{i \in \mathcal{N}_j} B_k[j, i] x_k^i \text{ and } e_k^j = \sum_{i \in \mathcal{N}_j} B_k[j, i] d_k^i.$$

Importantly note that since  $\sum_j d_k^j = 0$  and  $B_k$  is doubly stochastic, we also get,  $\sum_j e_k^j = 0$ .

This rewrite also allows to rearrange the projected gradient descent expression (Eq. 7, see Line 8, Algo. 1) as,

$$x_{k+1}^j = \mathcal{P}_{\mathcal{X}} \left[ \hat{v}_k^j - \alpha_k \left( \nabla f_j(v_k^j) - e_k^j \right) \right]. \quad (13)$$

This perspective of considering the perturbations as noise in the gradients is useful for analysis. It will be used to motivate our second algorithm, **FS** (Algorithm 4) in Section 5.

We note an important result about convergence of product of doubly stochastic weight matrices from [49]. This result helps us construct a bound (Lemma 1) on the maximum deviation between a model parameter estimate and the average estimate (over the network) at any iteration  $k$ . Next, in Lemma 2, we develop a bound on the distance between local estimates and the optimum (at iteration  $k + 1$ ) as a function of the distance in prior iteration  $k$ , function values and a constant dependent on the problem parameters (e.g.  $L, N, S$ ). Final convergence result, claiming convergence of local estimates to the optimum, is stated as Theorem 1 later in this section.

### Transition Matrix Analysis

We define transition matrix as the product of doubly stochastic weight matrices  $B_k$ ,

$$\Phi(k, s) = B_k B_{k-1} \dots B_{s+1} B_s, \quad (\forall k \geq s \geq 0).$$

We further use transition matrix convergence analysis from [49] to claim linear convergence of transition matrix entries  $\Phi(k, s)[j, i]$  to  $1/S$ . That is,

$$\left| \Phi(k, s)[i, j] - \frac{1}{S} \right| \leq \theta \beta^{k-s+1}, \quad (\forall k \geq s \geq 0)$$

where  $\theta = (1 - \frac{\rho}{4S^2})^{-2}$  and  $\beta = (1 - \frac{\rho}{4S^2})$  are constants that depend only on graph topology.<sup>5</sup>

### Disagreement Lemma

Lemma 1, below, referred to as the disagreement lemma provides a bound on the deviation of a estimate from average estimate over the network. Please recall the definition of the disagreement ( $\delta_k^j$ , Eq. 3) between local estimate ( $x_k^j$ ) and the average estimate ( $\bar{x}_k$ ).

**Lemma 1.** *Let iterates be generated by Algorithm 1, while Assumptions 1, 2 and 3 hold. Then,*

$$\max_{j \in \mathcal{V}} \|\delta_{k+1}^j\| \leq S\theta\beta^{k+1} \max_{i \in \mathcal{V}} \|x_0^i\| + S\theta(L + \Delta) \sum_{l=1}^k \beta^{k+1-l} \alpha_{l-1} + 2\alpha_k(L + \Delta)$$

□

Complete proofs cannot be included here due to space constraints. Proofs for all Lemmas and Theorems are included in appendix. The first term of the bound on the disagreement is the result of convex averaging of neighboring estimates in the consensus step. The second and the third terms are an artifact of the gradient descent steps performed by agents locally. And following our viewpoint of considering state perturbation as a gradient noise, we see random noise bound ( $\Delta$ ) appear in the second and third term. The first term of the bound on the disagreement shows linear decay (rate:  $\beta$ ) due to convex averaging. The third term decays with rate  $\mathcal{O}(1/k)$  since  $\alpha_k$  decays with rate  $\mathcal{O}(1/k)$ .

<sup>5</sup> Recall,  $\rho$  is the smallest nonzero entry in matrix  $B_k$  for any  $k$ .



### Iterate Lemma

We next develop a relationship between distance between estimate and its average at two consecutive time steps. It characterizes the correspondence between the distance of an iterate from a point  $y \in \mathcal{X}$ ,  $\|x_k^j - y\|$ , the function value at the iterate average  $\bar{x}_k$ , and the function value at  $y$ .

**Lemma 2.** *Let Assumptions 1, 2, 3 hold. The iterates generated by Algorithm 1 satisfy the following relationship for all  $y \in \mathcal{X}$ ,*

$$\begin{aligned}\eta_{k+1}^2 &\leq (1 + F_k) \eta_k^2 - 2\alpha_k (f(\bar{x}_k) - f(y)) + H_k, \text{ where} \\ \eta_k^2 &= \sum_{j=1}^S \|x_k^j - y\|^2, \quad F_k = \alpha_k N \left( \max_{j \in \mathcal{V}} \|\delta_k^j\| + \alpha_k \Delta \right), \text{ and} \\ H_k &= 2S\alpha_k \max_{j \in \mathcal{V}} \|\delta_k^j\| (L + N/2 + \Delta) + S\alpha_k^2 [N\Delta + (L + \Delta)^2].\end{aligned}$$

□

Note that if  $y = x^* \in \mathcal{X}^*$ , then (at iteration  $k$ ),

- $\eta_k^2$  represents the squared distance between parameter estimates and optimal choice of model parameters.
- $F_k$  is a positive sequence.
- $f(\bar{x}_k) - f(x^*) (\geq 0)$  represents the optimality gap for the current model parameter estimates.
- $H_k$  is a positive sequence dependent on problem parameters and the bound on maximum deviation  $\max_{j \in \mathcal{V}} \|\delta_k^j\|$ .

If  $F_k$  had been a negative sequence, one could argue that as  $H_k$  became smaller, the distance of estimates from the optimum would keep decreasing due to positive optimality gap (providing convergence to the optimum). However, since  $F_k \geq 0$ , we need special machinery that considers evolution of both sequences  $\{F_k\}$  and  $\{H_k\}$  with respect to iteration  $k$ .

### Convergence of Algorithm 1

We first state the deterministic version of a known result on the convergence of non-negative almost supermartingales.

**Lemma 3.** [50]. *Let  $\{E_k\}, \{G_k\}$  and  $\{H_k\}$ , be non-negative, real sequences and let  $\{F_k\}$  be a deterministic, real sequence. Assume that  $\sum_{k=0}^{\infty} F_k < \infty$ , and  $\sum_{k=0}^{\infty} H_k < \infty$  and*

$$E_{k+1} \leq (1 + F_k)E_k - G_k + H_k.$$

*Then, the sequence  $\{E_k\}$  converges to a non-negative real number and  $\sum_{k=0}^{\infty} G_k < \infty$ .*

Note the similarities in the expressions in Lemma 2 and Lemma 3. We use the result  $\sum_k G_k < \infty$  from Lemma 3 to prove asymptotic convergence of the iterate average  $(\bar{x}_k)$  to the optimum  $(x^*)$  in function value (Eq. 2), i.e.  $f(\bar{x}_k) \rightarrow f(x^*)$  asymptotically.

The convergence result is formally stated in Theorem 1. The result states that all local estimates converge to the iterate average  $(x_k^j \rightarrow \bar{x}_k)$  asymptotically) and the iterate average converges to an optimum  $(\bar{x}_k \rightarrow x^*)$  asymptotically).

**Theorem 1.** *Let Assumptions 1, 2, and 3 hold with  $\mathcal{X}^*$  being a nonempty bounded set. Also assume a diminishing step size rule presented in Eq. 4. Then, the iterate average  $(\bar{x}_k)$  converges to an optimum in  $\mathcal{X}^*$  and sequence of iterates  $\{x_k^j\}$ , generated by Algorithm 1, converge to the iterate average  $\bar{x}_k$  asymptotically.*

**Proof Sketch:** Lemma 3 is central to the proof of Theorem 1. We consider the iterate inequality set up in Lemma 2 (Iterate Lemma) and set  $y = x^* \in \mathcal{X}^*$  and observe that it has the same structure as given in Lemma 3.  $E_k = \eta_k^2$ ,  $G_k = 2\alpha_k (f(\bar{x}_k) - f(x^*))$ , and  $F_k, H_k$  are as defined in Lemma 2.

We first need to show that  $\sum_k F_k < \infty$  and  $\sum_k H_k < \infty$  for Lemma 3 to be applicable. We do that in two steps first by proving that  $\sum_k \alpha_k \max_j \|\delta_k^j\| < \infty$  and then observing  $\sum_k \alpha_k^2 < \infty$  to prove  $\sum_k F_k < \infty$  and  $\sum_k H_k < \infty$  (see detailed proof in appendix, Section B.3).

Lemma 3 directly gives us that  $\lim_{k \rightarrow \infty} \eta_k^2$  exists and is finite, and  $\sum_k \alpha_k (f(\bar{x}_k) - f(x^*)) < \infty$ . We further use the fact that  $\sum_k \alpha_k = \infty$ , to get  $\liminf_{k \rightarrow \infty} f(\bar{x}_k) = f(x^*)$ . Due to the continuity of  $f(x)$ , we can then show that the sequence of iterate average enters the optimal set  $\mathcal{X}^*$  infinitely often at large  $k$ . We then use the fact that  $\lim_{k \rightarrow \infty} \eta_k^2$  in conjunction to show that the limit has to be zero. This gives convergence of iterate average to the optimum. Convergence of local estimates to the iterate average can be easily proved by using Lemma 1.

Detailed proof is included in appendix (Section B.3).  $\square$

## 4.2 RSS-LB Convergence Analysis

The convergence analysis of RSS-LB algorithm is very similar to that of RSS-GB algorithm. We begin by rewriting the information fusion expression (Line 7, Algorithm 3). The update is rewritten as the sum of fusion of true states,  $\hat{v}_k^j$ , and the fusion of random perturbations,  $e_k^j$ , as follows,

$$\begin{aligned} v_k^j &= \sum_{i \in \mathcal{N}_j} B_k[j, i] w_k^{i,j} = \hat{v}_k^j + \alpha_k e_k^j, \\ \hat{v}_k^j &= \sum_{i \in \mathcal{N}_j} B_k[j, i] x_k^i \text{ and } e_k^j = \sum_{i \in \mathcal{N}_j} B_k[j, i] d_k^{i,j}. \end{aligned} \quad (14)$$

This rewrite allows to rearrange the projected gradient descent expression (Eq. 11, Line 8, Algo. 3) as,

$$x_{k+1}^j = \mathcal{P}_{\mathcal{X}} \left[ \hat{v}_k^j - \alpha_k \left( \nabla f_j(v_k^j) - e_k^j \right) \right]. \quad (15)$$

This perspective of considering the perturbations as noise in the gradients simplifies analysis.

Importantly, as described in RSS-LB algorithm (Assumption 4, Section 3), we construct  $d_k^{i,j}$  such that the effective sum of random perturbations is zero,  $e_k^j = 0$ . Note that if the doubly stochastic matrix  $B_k$  is constructed such that each non-diagonal, non-zero entry is exactly  $\rho$ , then Assumption 4 is automatically satisfied for any locally balanced  $d_k^{i,j}$ . This makes analysis of RSS and RSS-LB similar.

As an example consider the rules for constructing  $B_k$  (from [48]),

$$B_k[j, i] = \begin{cases} 0, & \text{if } (j, i) \notin \mathcal{E} \\ \rho, & \text{if } (j, i) \in \mathcal{E} \\ 1 - |\mathcal{N}_j| \rho & \text{if } i = j \end{cases} \quad (16)$$

where,  $\rho = \frac{0.5}{\max_j |\mathcal{N}_j|}$  for all  $k$ . We can select any value for  $\rho$  as long as  $B_k[j, j] \geq 0$ , e.g.  $\rho = \frac{1}{S}$ . While computing  $e_k^j$  for any  $j$ , note that all  $\alpha_k d_k^{i,j}$  are multiplied by same off-diagonal weight  $\rho$ . This ensures that  $\sum_j e_k^j = \sum_j \sum_i \rho d_k^{i,j} = \rho \sum_i \sum_j d_k^{i,j} = 0$  (since  $\sum_j d_k^{i,j} = 0$  for all  $i$  and  $k$ ).

The proofs and the convergence results for RSS-LB are exactly the same as those for RSS algorithm (under an additional assumption, Assumption 4). We use convergence of product of doubly stochastic weight matrices to construct a bound (Lemma 4) on the maximum deviation between a model parameter estimate and the average estimate (over the network) at any iteration  $k$ . Next, in Lemma 5, we develop a bound on the distance between local estimates and the optimum (at iteration  $k + 1$ ) as a function of the distance in prior iteration  $k$ , function values and a constant dependent on the problem parameters (e.g.  $L, N, S$ ). Final convergence result, claiming convergence of local estimates to the optimum, is stated as Theorem 2 later in this section.

### Disagreement Lemma

**Lemma 4.** *Let iterates be generated by Algorithm 3, while Assumptions 1, 2, 3 and 4 hold. Then,*

$$\max_{j \in \mathcal{V}} \|\delta_{k+1}^j\| \leq S\theta \beta^{k+1} \max_{i \in \mathcal{V}} \|x_0^i\| + S\theta(L + \Delta) \sum_{l=1}^k \beta^{k+1-l} \alpha_{l-1} + 2\alpha_k (L + \Delta)$$

### Iterate Lemma

**Lemma 5.** *Let Assumptions 1, 2, 3 and 4 hold. The iterates generated by Algorithm 3 satisfy the following relationship for all  $y \in \mathcal{X}$ ,*

$$\begin{aligned}\eta_{k+1}^2 &\leq (1 + F_k) \eta_k^2 - 2\alpha_k (f(\bar{x}_k) - f(y)) + H_k, \text{ where} \\ \eta_k^2 &= \sum_{j=1}^S \|x_k^j - y\|^2, \quad F_k = \alpha_k N \left( \max_{j \in \mathcal{V}} \|\delta_k^j\| + \alpha_k \Delta \right), \text{ and} \\ H_k &= 2S\alpha_k \max_{j \in \mathcal{V}} \|\delta_k^j\| (L + N/2 + \Delta) + S\alpha_k^2 [N\Delta + (L + \Delta)^2].\end{aligned}$$

### Convergence of Algorithm 3

**Theorem 2.** *Let Assumptions 1, 2, 3 and 4 hold with  $\mathcal{X}^*$  being a nonempty bounded set. Also assume a diminishing step size rule presented in Eq. 4. Then, the iterate average  $(\bar{x}_k)$  converges to an optimum in  $\mathcal{X}^*$  and sequence of iterates  $\{x_k^j\}$ , generated by Algorithm 3, converges to the iterate average  $\bar{x}_k$  asymptotically.*

## 5 Special Case - Function Sharing (FS) [1]

As discussed earlier, gradient descent update for the RSS algorithm can be rewritten to incorporate the perturbation term into the gradient (Eq. 13). This simulates a scenario where the gradient computation is erroneous (and the error is  $-e_k^j$ ). However, we note that  $\sum_j e_k^j = 0$ . Now consider that the perturbations added to the state in Eq. 5 are *state dependent*, i.e. for the same  $\hat{v}_k^j$  we have the same  $e_k^j$  in every execution. Under this added assumption of *state dependent randomness*, we can construct a function  $p_j(x)$  such that  $\nabla p_j(v_k^j) = -e_k^j$ , for all  $k$ . Clearly, adding  $-e_k^j$  to the gradient is effectively perturbing the objective function with  $p_j(x)$ . Since  $\sum_j e_k^j = 0$  for all  $k$ , we get  $\sum_j \nabla p_j(v_k^j) = 0$  for all  $v_k^j$ .

This implies that, the state perturbation strategy (in RSS) with an addition condition of *state dependent randomness*, can be achieved by modifying  $f_j(x)$  with  $p_j(x)$  as follows,

$$\hat{f}_j(x) = f_j(x) + p_j(x), \quad (17)$$

such that  $\sum_j p_j(x) = 0$ . It implies  $f(x) = \sum_j f_j(x) = \sum_j \hat{f}_j(x)$ . The function obfuscation step is followed by DGD algorithm (Algorithm 0). We call this algorithm the function sharing approach to privacy preserving optimization. Formally, the algorithm is stated as Algorithm 4.

### Generating $p_j(x)$

Constructing correlated perturbation functions  $p_j(x)$  in a decentralized manner is easy and follows directly from the process in Section 3.2. Instead of exchanging random numbers, agents exchange randomly generated functions ( $s^{i,j}(x)$ ) to generate  $p_j(x)$  as in Eq. 8.

---

#### Algorithm 4 Function Sharing (FS) Algorithm

---

- 1: Each agent  $j$  computes correlated noise function,  $p_j(x)$
  - 2: Each agent  $j$  computes obfuscated objective,  $\hat{f}_j(x)$ ,  
 $\hat{f}_j(x) = f_j(x) + p_j(x)$ .
  - 3: **for**  $k = 1$  to MAX\_ITER **do**
  - 4:   Information Fusion  
 $v_k^j = \sum_{i \in \mathcal{N}_j} B_k[j, i] x_k^i$
  - 5:   Projected Gradient Descent  
 $x_{k+1}^j = \mathcal{P}_{\mathcal{X}} \left[ v_k^j - \alpha_k \nabla \hat{f}_j(v_k^j) \right]$
  - 6: **end for**
-

## Convergence of Algorithm 4

The analysis for Algorithm 1 holds with the observation that the gradient of  $p_j(x)$  at  $v_k^j$  serves the same role as  $-e_k^j$ . We can claim convergence of iterates to the optimal set. Note that the obfuscated function  $\hat{f}_i(x)$  need not be convex, however, its sum  $f(x) = \sum_j \hat{f}_j(x)$  is convex. This effectively implies that we are minimizing convex sum of non-convex functions. The analysis here extends the standard distributed convex optimization literature where each function  $f_i(x)$  is assumed to be convex. A similar generalization, albeit with additional assumption of strong convexity of  $f(x)$  and without gradient boundedness condition (Assumption 2(A)), was proven using Lyapunov stability arguments in [42].

## 6 Discussion

### 6.1 Privacy Motivations for RSS and RSS-LB

We prove in our companion work [1], that FS algorithm provides privacy against an adversary that can corrupt at most  $f$  nodes if the underlying graph topology has a vertex connectivity of  $f + 1$  or higher.

RSS and RSS-LB algorithms provide obvious improvement in privacy (over DGD algorithm) by randomly perturbing the shared state estimate. Moreover, since the random numbers can be drawn from any distribution (possibly time varying), statistical techniques become less useful. We will present detailed privacy analysis in a future report.

### 6.2 Machine Learning Example

We will first observe that distributed classification (e.g. handwritten digit classification) problem fits the structure of Problem 1. Next we use FS algorithm to perform privacy preserving learning. For FS strategy we need to generate noise functions. However, the noise functions are not easy to construct and there is no general template that works for all problems. We will then see RSS algorithm that is general (applies to a large class of machine learning problems) and easy to implement.

Each agent  $i$  owns labeled training dataset  $D_i$ , comprising of samples of type  $(x_j, y_j)$ . We intend to learn a linear logistic regression model for classification. The local loss function associated with agent  $i$  is defined as,

$$f_i(w) = \sum_{(x_l, y_l) \in D_i} \log(1 + (-y_l w^T x_l)) + (\lambda/2S) \|w\|_2^2$$

where,  $\lambda$  is the weight associated with regularization term. Distributed learning problem (Problem 1) is rewritten as,

$$\min_w \left( \sum_{i=1}^S \left[ \sum_{(x_l, y_l) \in D_i} \log(1 + (-y_l w^T x_l)) \right] + \frac{\lambda}{2} \|w\|_2^2 \right)$$

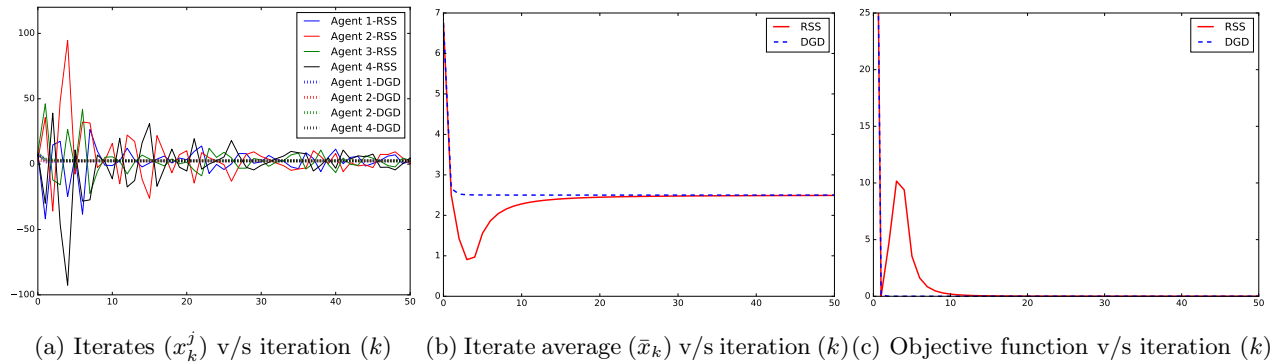


Figure 1: Simulation results for a simple example.

The logistic loss function is strongly convex, and the gradient is  $L$ -Lipschitz and norm bounded for finite  $w$ . **FS Algorithm:** We need to design  $s^{i,j}(x)$  carefully to ensure that  $p_i(x)$  hides  $f_i(x)$ . Consider that each agent creates spurious samples ( $E_{i,j}$ , e.g. incorrectly labeled samples) and shares the function corresponding to them, i.e. agent  $i$  shares  $s^{i,j}(x)$  with agent  $j$ , where,  $b_{i,j}$  is bounded random number ( $|b_{i,j}| \leq \lambda/2S$ ).

$$s^{i,j}(x) = \sum_{l \in E_{i,j}} \log(1 + (-y_l w^T x_l)) + b_{i,j} \|w\|_2^2,$$

The obfuscated objective function  $\hat{f}_i(x)$  becomes,

$$\hat{f}_i(x) = \sum_{l \in \tilde{D}_i} \log(1 + (-y_l w^T x_l)) + \left( \frac{\lambda}{2S} + \tilde{b}_i \right) \|w\|_2^2 - \sum_{l \in \cup_{j \in \mathcal{N}_i} E_{i,j}} \log(1 + (-y_l w^T x_l))$$

where  $\tilde{D}_i = D_i + \cup_{j \in \mathcal{N}_i} E_{j,i}$  and  $\tilde{b}_i = \sum_{j \in \mathcal{N}_i} b_{j,i} - \sum_{j \in \mathcal{N}_i} b_{i,j}$ .  $s^{i,j}(x)$  generated using spurious data may work in this example, however, it is clear that learning via FS algorithm is not straightforward.

**RSS Algorithm:** The RSS algorithm involves using randomized perturbations to improve privacy. Structured random numbers can be easily generated using Algorithm 2.

**Simple Example:** We construct a simple example with four agents connected in a cycle graph (every agent has two neighbors). The local objective functions are  $f_i(x) = (x - i)^2$ , where  $i = \{1, 2, 3, 4\}$ . This problem satisfies all the assumptions that we use. We consider noise with bound,  $\Delta = 1000$  and  $\mathcal{X} = [-200, 200]$ . The DGD converges quickly, takes around 20, while, RSS takes around 50 iterations. Both algorithms accurately calculate minimum. Note that the high variance seen in iterates (Figure 1a) is not seen in the iterate average (Figure 1b). This happens because the perturbations being add up to zero (at every  $k$ ). Notice the objective functions does not decrease monotonically (Figure 1c). This is to be expected since the noise may nudge the iterates in random direction.

## 7 Conclusion

In this paper, we present two privacy preserving algorithms for distributed learning over networks. RSS algorithm involves perturbations that are balanced over the network via the use of Algorithm 2. RSS-LB algorithm uses perturbations that are locally balanced. We prove that state perturbation with correlated noise allows us to learn accurate models in deterministic sense. We show that we can recover Function Sharing strategy (cf. [1]) from Algorithm 1, by selecting noise that is dependent on estimates. FS strategy uses correlated (among agents) noise functions are used to obfuscate private objective functions to protect their privacy.

## References

- [1] S. Gade and N. H. Vaidya, "Private learning on networks," *arXiv preprint arXiv:1612.05236*, 2016.
- [2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.
- [3] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan, "Mlbase: A distributed machine-learning system.," in *CIDR*, vol. 1, pp. 2–1, 2013.
- [4] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Advances in Neural Information Processing Systems 27*, pp. 19–27, Curran Associates, Inc., 2014.
- [5] M. Li, L. Zhou, Z. Yang, A. Li, and F. Xia, "Parameter Server for Distributed Machine Learning," *Big Learning Workshop*, pp. 1–10, 2013.
- [6] I. Cano, M. Weimer, D. Mahajan, C. Curino, and G. M. Fumarola, "Towards geo-distributed machine learning," *arXiv preprint arXiv:1603.09035*, 2016.

- [7] J. Konečný, B. McMahan, and D. Ramage, “Federated optimization: Distributed optimization beyond the datacenter,” *arXiv preprint arXiv:1511.03575*, 2015.
- [8] Y. Nesterov, “Efficiency of coordinate descent methods on huge-scale optimization problems,” *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.
- [9] C. Singh, A. Nedić, and R. Srikant, “Random Block-Coordinate Gradient Projection Algorithms,” in *53rd IEEE Conference on Decision and Control*, pp. 185–190, 2014.
- [10] A. Nedić and A. Ozdaglar, “On the rate of convergence of distributed subgradient methods for multi-agent optimization,” *Proceedings of the IEEE Conference on Decision and Control*, vol. 54, no. 1, pp. 4711–4716, 2007.
- [11] M. Rabbat and R. Nowak, “Distributed optimization in sensor networks,” in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pp. 20–27, ACM, 2004.
- [12] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for machine learning*. MIT Press, 2012.
- [13] A. Nedić and A. Olshevsky, “Distributed Optimization Over Time-Varying Directed Graphs,” *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.
- [14] C. N. Hadjicostis, N. H. Vaidya, and A. D. Domínguez-García, “Robust distributed average consensus via exchange of running sums,” *IEEE Transactions on Automatic Control*, vol. 61, no. 6, pp. 1492–1507, 2016.
- [15] A. Nedić, “Asynchronous broadcast-based convex optimization over a network,” *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1337–1351, 2011.
- [16] J. Liu and S. J. Wright, “Asynchronous stochastic coordinate descent: Parallelism and convergence properties,” *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 351–376, 2015.
- [17] E. Wei and A. Ozdaglar, “On the  $O(1/k)$  convergence of asynchronous distributed alternating direction method of multipliers,” in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pp. 551–554, IEEE, 2013.
- [18] A. Agarwal and J. C. Duchi, “Distributed delayed stochastic optimization,” in *Advances in Neural Information Processing Systems*, pp. 873–881, 2011.
- [19] S. S. Ram, A. Nedić, and V. V. Veeravalli, “Distributed stochastic subgradient projection algorithms for convex optimization,” *Journal of optimization theory and applications*, vol. 147, no. 3, pp. 516–545, 2010.
- [20] L. Su and N. H. Vaidya, “Fault-tolerant multi-agent optimization: Optimal iterative distributed algorithms,” in *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pp. 425–434, ACM, 2016.
- [21] R. Shokri and V. Shmatikov, “Privacy-Preserving Deep Learning,” *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, pp. 1310–1321, 2015.
- [22] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” *arXiv preprint arXiv:1607.00133*, 2016.
- [23] E. A. Abbe, A. E. Khandani, and A. W. Lo, “Privacy-preserving methods for sharing financial risk exposures,” *The American Economic Review*, vol. 102, no. 3, pp. 65–70, 2012.
- [24] M. Alizadeh, T.-H. Chang, and A. Scaglione, “Grid integration of distributed renewables through coordinated demand response,” in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 3666–3671, IEEE, 2012.

- [25] F. Pasqualetti, F. Dörfler, and F. Bullo, “Cyber-physical security via geometric control: Distributed monitoring and malicious attacks,” in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 3418–3425, IEEE, 2012.
- [26] H. B. McMahan, E. Moore, D. Ramage, *et al.*, “Federated learning of deep networks using model averaging,” *arXiv preprint arXiv:1602.05629*, 2016.
- [27] P. C. Weeraddana, G. Athanasiou, C. Fischione, and J. S. Baras, “Per-se privacy preserving solution methods based on optimization,” in *Proceedings of the 52nd IEEE Conference on Decision and Control (CDC)*, pp. 206–211, 2013.
- [28] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.
- [29] S. Goldwasser, “Multi party computations: past and present,” in *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pp. 1–6, ACM, 1997.
- [30] B. Pinkas, “Cryptographic techniques for privacy-preserving data mining,” *ACM Sigkdd Explorations Newsletter*, vol. 4, no. 2, pp. 12–19, 2002.
- [31] Y. Hong, J. Vaidya, N. Rizzo, and Q. Liu, “Privacy preserving linear programming,” *arXiv preprint arXiv:1610.02339*, 2016.
- [32] A. Bednarz, *Methods for two-party privacy-preserving linear programming*. PhD thesis, 2012.
- [33] Z. Huang, S. Mitra, and N. Vaidya, “Differentially private distributed optimization,” in *Proceedings of the 2015 International Conference on Distributed Computing and Networking*, p. 4, ACM, 2015.
- [34] E. Nozari, P. Tallapragada, and J. Cortés, “Differentially private distributed convex optimization via functional perturbation,” *arXiv preprint arXiv:1512.00369*, 2015.
- [35] S. Han, U. Topcu, and G. J. Pappas, “Differentially private distributed constrained optimization,” *IEEE Transactions on Automatic Control*, vol. PP, no. 99, pp. 1–1, 2016.
- [36] J. Hamm, Y. Cao, and M. Belkin, “Learning privately from multiparty data,” in *Proceedings of The 33rd International Conference on Machine Learning*, pp. 555–563, 2016.
- [37] P. C. Weeraddana, G. Athanasiou, M. Jakobsson, C. Fischione, and J. S. Baras, “On the privacy of optimization approaches,” *arXiv preprint arXiv:1210.3283*, 2012.
- [38] O. L. Mangasarian, “Privacy-preserving horizontally partitioned linear programs,” *Optimization Letters*, vol. 6, no. 3, pp. 431–436, 2012.
- [39] C. Wang, K. Ren, and J. Wang, “Secure and practical outsourcing of linear programming in cloud computing,” in *INFOCOM, 2011 Proceedings IEEE*, pp. 820–828, IEEE, 2011.
- [40] J. Dreier and F. Kerschbaum, “Practical privacy-preserving multiparty linear programming based on problem transformation,” in *Third IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT’11)*, 2011.
- [41] Y. Lou, L. Yu, and S. Wang, “Privacy preservation in distributed subgradient optimization algorithms,” *arXiv preprint arXiv:1512.08822*, 2015.
- [42] K. Kvaternik and L. Pavel, “Lyapunov analysis of a distributed optimization scheme,” in *Network Games, Control and Optimization (NetGCooP), 2011 5th International Conference on*, pp. 1–5, IEEE, 2011.
- [43] S. Gade and N. Vaidya, “Distributed optimization of convex sum of non-convex functions,” *arXiv preprint arXiv:1608.05401*, 2016.

- [44] S. Gade and N. Vaidya, “Distributed optimization for client-server architecture with negative gradient weights,” *arXiv preprint arXiv:1608.03866*, 2016.
- [45] L. Xiao, S. Boyd, and S. Lall, “Distributed average consensus with time-varying metropolis weights,” *Automatica*, 2006.
- [46] L. Xiao, S. Boyd, and S.-J. Kim, “Distributed average consensus with least-mean-square deviation,” *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, 2007.
- [47] V. Blondel, J. M. Hendrickx, A. Olshevsky, J. Tsitsiklis, *et al.*, “Convergence in multiagent coordination, consensus, and flocking,” in *IEEE Conference on Decision and Control*, vol. 44, p. 2996, IEEE; 1998, 2005.
- [48] J. Koshal, A. Nedić, and U. V. Shanbhag, “Distributed algorithms for aggregative games on graphs,” *Operations Research*, vol. 64, no. 3, pp. 680–704, 2016.
- [49] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, “Distributed subgradient methods and quantization effects,” in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 4177–4184, IEEE, 2008.
- [50] H. Robbins and D. Siegmund, “A convergence theorem for non negative almost supermartingales and some applications,” in *Herbert Robbins Selected Papers*, pp. 111–135, Springer, 1985.
- [51] D. P. Bertsekas, A. Nedić, A. E. Ozdaglar, *et al.*, *Convex analysis and optimization*. Athena Scientific, 2003.
- [52] J. A. Fax and R. M. Murray, “Information flow and cooperative control of vehicle formations,” *IEEE transactions on automatic control*, vol. 49, no. 9, pp. 1465–1476, 2004.
- [53] A. Nedić and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” *Automatic Control, IEEE Transactions on*, vol. 54, no. 1, pp. 48–61, 2009.



# Appendix

## A Notation Summary

Summary of symbols and constants used in the analysis is presented in Table 1

Symbol	Description
$\mathcal{V}$	Set of Agents (Nodes)
$\mathcal{E}$	Set of Communication Links (Edges)
$\mathcal{G}(\mathcal{V}, \mathcal{E})$	Graph with Nodes $\mathcal{V}$ and Edges $\mathcal{E}$
$S$	Number of Agents (Nodes) $S =  \mathcal{V} $
$\mathcal{N}_j$	Neighborhood set of agent $j$
$f(x)$	Objective function
$f_i(x)$	Objective function associated with Agent $i$
$D$	Dimension of the optimization problem (size of vector $x$ )
$\mathcal{X}$	Feasible Set (Model Parameters)
$\mathcal{X}^*$	Optimal Set (Model Parameters)
$x_k^j$	Model parameter estimate (iterate/state) with agent $j$ at time $k$
$\bar{x}_k$	Average model (over the network) at time $k$
$\delta_k^j$	Disagreement between agent $j$ and iterate-average, $\delta_k^j = x_k^j - \bar{x}_k$
$L$	Gradient bound, $\ \nabla f_i(x)\  \leq L$
$N$	Lipschitz constant for $\nabla f_i(x)$
$B_k$	Doubly stochastic matrix
$\rho$	Smallest non-zero entry in matrix $B_k$
$\alpha_k$	Learning step-size
$d_k^j$	Correlated random perturbation
$w_k^j$	Perturbed local model estimate
$\Delta$	Random perturbation bound, $\ d_k^j\  \leq \Delta$
$v_k^j$	Fused (perturbed) model estimate, Eq. 6
$\hat{v}_k^j$	Fused (true) model estimate, Eq. 12
$c_k^j$	Fused perturbation, Eq. 12
$\Phi(k, s)$	Transition Matrix
$\theta$	Transition matrix convergence parameter, $\theta = (1 - \frac{\rho}{4S^2})^{-2}$
$\beta$	Transition matrix convergence parameter, $\beta = (1 - \frac{\rho}{4S^2})$
$\eta_k^2$	Distance between iterate and optimum, $\eta_k^2 = \sum_{j=1}^S \ x_k^j - y\ ^2$
$\xi_k^2$	Distance between fused iterate and optimum, $\xi_k^2 = \sum_{j=1}^S \ \hat{v}_k^j - y\ ^2$
$F_k$	$F_k = \alpha_k N \left( \max_{j \in \mathcal{V}} \ \delta_k^j\  + \alpha_k \Delta \right)$
$H_k$	$H_k = 2\alpha_k S \max_{j \in \mathcal{V}} \ \delta_k^j\  (L + N/2 + \Delta) + \alpha_k^2 S [N\Delta + (L + \Delta)^2]$

Table 1: Parameters and Constants

## B Proofs

We first note two important results from literature. The first result relates to convergence of non-negative sequences and the second result describes the convergence of transition matrix to  $\frac{1}{S} \mathbb{1}\mathbb{1}^T$ .

**Lemma 4** (Lemma 3.1, [19]). *Let  $\{\zeta_k\}$  be a non-negative scalar sequence. If  $\sum_{k=0}^{\infty} \zeta_k < \infty$  and  $0 < \beta < 1$ , then  $\sum_{k=0}^{\infty} \left( \sum_{j=0}^k \beta^{k-j} \zeta_j \right) < \infty$ .*

**Lemma 5** (Corollary 1, [49]). *Let Assumption 3 hold then,*

1.  $\lim_{k \rightarrow \infty} \Phi(k, s) = \frac{1}{S} \mathbb{1} \mathbb{1}^T$  for all  $s \geq 0$ .
2.  $|\Phi(k, s)[i, j] - \frac{1}{S}| \leq \theta \beta^{k-s+1}$  for all  $k \geq s \geq 0$ , where  $\theta = (1 - \frac{\rho}{4S^2})^{-2}$  and  $\beta = (1 - \frac{\rho}{4S^2})$ .

The well known non-expansive property (cf. [51]) of Euclidean projection onto a non-empty, closed, convex set  $\mathcal{X}$ , is represented by the following inequality,  $\forall x, y \in \mathbb{R}^D$ ,

$$\|\mathcal{P}_{\mathcal{X}}[x] - \mathcal{P}_{\mathcal{X}}[y]\| \leq \|x - y\|. \quad (18)$$

This non-expansive property of the projection operator will be used extensively in our analysis.

The proofs in this report follow the structure similar to [19], with the key difference being in proof for Lemma 2 and Theorem 1.

## B.1 Proof of Lemma 1 (Disagreement Lemma)

*Proof.* Define for all  $j \in \mathcal{V}$  and all  $k$ ,

$$z_{k+1}^j = x_{k+1}^j - \sum_{i=1}^S B_k[j, i] x_k^i \quad (19)$$

We then unroll the iterations to get  $x_{k+1}^j$  as a function of  $x_k^i$ ,  $x_{k-1}^i$ ,  $x_k^i$  and doubly stochastic weight matrix at current and previous iteration.

$$x_{k+1}^j = z_{k+1}^j + \sum_{i=1}^S \left[ B_k[j, i] \left( z_k^i + \sum_{l=1}^S B_{k-1}[i, l] x_{k-1}^l \right) \right]$$

We perform the above mentioned unrolling successively and use the definition of transition matrix  $\Phi(k, s)$ ,

$$x_{k+1}^j = z_{k+1}^j + \sum_{i=1}^S \Phi(k, 0)[j, i] x_0^i + \sum_{l=1}^k \left[ \sum_{i=1}^S \Phi(k, l)[j, i] z_l^i \right]. \quad (20)$$

Note that following the definition of transition matrix,  $\Phi(0, 0) = B_0$ . We verify the expression for  $k = 0$ , and we get the relationship  $x_1^j = z_1^j + \sum_{i=1}^S \Phi(0, 0)[j, i] x_0^i$ .

We can rewrite the relation for iterate average,  $\bar{x}_k$ , and use doubly stochastic nature of  $B_k$  ( $\sum_{j=1}^S B_k[j, i] = 1$ ) to get,

$$\begin{aligned} \bar{x}_{k+1} &= \frac{1}{S} \sum_{j=1}^S x_{k+1}^j = \frac{1}{S} \sum_{j=1}^S \left( \sum_{i=1}^S B_k[j, i] x_k^i + z_{k+1}^j \right) = \frac{1}{S} \left( \sum_{i=1}^S \left( \sum_{j=1}^S B_k[j, i] \right) x_k^i + \sum_{j=1}^S z_{k+1}^j \right), \\ &= \bar{x}_k + \frac{1}{S} \sum_{j=1}^S z_{k+1}^j = \bar{x}_0 + \frac{1}{S} \sum_{l=1}^{k+1} \sum_{j=1}^S z_l^j. \end{aligned} \quad (21)$$

Using relations for  $\bar{x}_{k+1}$  (Eq. 21) and  $x_{k+1}^j$  (Eq. 20) to get an expression for the disagreement. We further use the properties of norms ( $\|\sum a\| \leq \sum \|a\|$ ,  $\forall a$ ), to get,

$$\|x_{k+1}^j - \bar{x}_{k+1}\| \leq \sum_{i=1}^S \left| \frac{1}{S} - \Phi(k, 0)[j, i] \right| \|x_0^i\| + \sum_{l=1}^k \sum_{j=1}^S \left| \frac{1}{S} - \Phi(k, l)[j, i] \right| \|z_l^j\| + \|z_{k+1}^j\| + \frac{1}{S} \sum_{j=1}^S \|z_{k+1}^j\|. \quad (22)$$

We use Lemma 5 to bound terms of type  $\left| \frac{1}{S} - \Phi(k, l) \right|$  and  $\max_{i \in \mathcal{V}} \|x_0^i\|$  to bound  $\|x_0^i\|$ , to get,

$$\|x_{k+1}^j - \bar{x}_{k+1}\| \leq S\theta\beta^{k+1} \max_{i \in \mathcal{V}} \|x_0^i\| + \theta \sum_{l=1}^k \beta^{k+1-l} \sum_{i=1}^S \|z_l^i\| + \|z_{k+1}^j\| + \frac{1}{S} \sum_{j=1}^S \|z_{k+1}^j\|. \quad (23)$$

We now bound each of the norms  $\|z_k^j\|$ , using the fact that  $v_k^j \in \mathcal{X}$ , the non-expansive property of projection operator (see Eq. 18), and Assumption 2,

$$\begin{aligned}
\|z_{k+1}^j\| &= \|\mathcal{P}_{\mathcal{X}}[\hat{v}_k^j - \alpha_k (\nabla f_j(v_k^j) - e_k^j)] - \hat{v}_k^j\| \\
&\leq \alpha_k \|\nabla f_j(v_k^j) - e_k^j\| \\
&\leq \alpha_k (\|\nabla f_j(v_k^j)\| + \|e_k^j\|) && \text{Triangle inequality} \\
&\leq \alpha_k (L + \Delta) && \|e_k^j\| \leq \Delta
\end{aligned} \tag{24}$$

Note that we used the boundedness of perturbation  $e_k^j$  to obtain the above relation. Since the random perturbations ( $d_k^j$  or  $d_k^{i,j}$ ) are bounded by  $\Delta$ , their convex combination is also bounded by  $\Delta$ . Next, recall the definition of disagreement between an iterate and its average from Eq. 3. Combining Eq. 23 and Eq. 24,

$$\max_{j \in \mathcal{V}} \|x_{k+1}^j - \bar{x}_{k+1}\| = \max_{j \in \mathcal{V}} \|\delta_{k+1}^j\| \leq S\theta\beta^{k+1} \max_{i \in \mathcal{V}} \|x_0^i\| + S\theta(L + \Delta) \sum_{l=1}^k \beta^{k+1-l} \alpha_{l-1} + 2\alpha_k (L + \Delta)$$

□

## B.2 Proof of Lemma 2 (Iterate Lemma)

*Proof.* To simplify analysis, we adopt the following notation,

$$\eta_k^2 = \sum_{j=1}^S \|x_k^j - y\|^2, \quad \xi_k^2 = \sum_{j=1}^S \|\hat{v}_k^j - y\|^2. \tag{25}$$

Every iteration in the algorithm involves two steps, a) information fusion using consensus step and b) projected gradient descent on local objective function. The fused state ( $v_k^j$ ) is obtained from the neighbor states using Eq. 6. The fused state is further updated based on projected gradient descent given by Eq. 7.

We have  $\mathcal{P}_{\mathcal{X}}[y] = y$  (for all  $y \in \mathcal{X}$  where  $\mathcal{X}$  is non-empty, closed and convex set). Using the non-expansive property of the projection operator (Eq. 18), and the projected gradient descent expression in Eq. 13 we get,

$$\begin{aligned}
\|x_{k+1}^j - y\|^2 &= \|\mathcal{P}_{\mathcal{X}}[\hat{v}_k^j - \alpha_k (\nabla f_j(v_k^j) - e_k^j)] - y\|^2 \\
&\leq \|\hat{v}_k^j - \alpha_k (\nabla f_j(v_k^j) - e_k^j) - y\|^2
\end{aligned} \tag{26}$$

$$\leq \|\hat{v}_k^j - y\|^2 + \alpha_k^2 \|\nabla f_j(v_k^j) - e_k^j\|^2 - 2\alpha_k \left[ \nabla f_j(v_k^j) - e_k^j \right]^T (\hat{v}_k^j - y) \tag{27}$$

Adding the inequalities in Eq. 27 for all agents  $j = 1, 2, \dots, S$  and further using the boundedness of gradients and perturbations we get the following inequality,

$$\eta_{k+1}^2 \leq \xi_k^2 + \alpha_k^2 S(L + \Delta)^2 - 2\alpha_k \sum_{j=1}^S (\nabla f_j(v_k^j) - e_k^j)^T (\hat{v}_k^j - y) \tag{28}$$

We use consensus relationship used for information fusion (Eq. 12). We know that in D-dimension the consensus step can be rewritten using Kronecker product of D-dimension identity matrix ( $I_D$ ) and the doubly stochastic weight matrix ( $B_k$ ) [52]. We compare norms of both sides (2-norm),

$$\hat{\mathbf{v}}_k = (I_D \otimes B_k) \mathbf{x}_k \tag{29}$$

$$\begin{aligned}
\hat{\mathbf{v}}_k - \mathbf{y} &= (I_D \otimes B_k)(\mathbf{x}_k - \mathbf{y}) \\
\|\hat{\mathbf{v}}_k - \mathbf{y}\|_2^2 &= \|(I_D \otimes B_k)(\mathbf{x}_k - \mathbf{y})\|_2^2 \\
&\leq \|(I_D \otimes B_k)\|_2^2 \|\mathbf{x}_k - \mathbf{y}\|_2^2
\end{aligned} \tag{30}$$

We use the property of eigenvalues of Kronecker product of matrices. The eigenvalues of  $I_D \otimes B_k$  are essentially  $D$  copies of eigenvalues of  $B_k$ . Since  $B_k$  is a doubly stochastic matrix, its eigenvalues are upper bounded by 1. Clearly,  $\|(I_D \otimes B_k)\|_2^2 = \lambda_{\max}((I_D \otimes B_k)^\dagger(I_D \otimes B_k)) \leq 1$ . This follows from the fact that  $(I_D \otimes B_k)^\dagger(I_D \otimes B_k)$  is a doubly stochastic matrix since product of two doubly stochastic matrices is also doubly stochastic.<sup>6</sup>

$$\xi_k^2 = \|\hat{\mathbf{v}}_k - \mathbf{y}\|_2^2 \leq \|(\mathbf{x}_k - \mathbf{y})\|_2^2 = \eta_k^2 \quad (31)$$

Merging the inequalities in Eq. 31 and Eq. 28, we get,

$$\eta_{k+1}^2 \leq \eta_k^2 + \alpha_k^2 S(L + \Delta)^2 - 2\alpha_k \underbrace{\sum_{j=1}^S (\nabla f_j(v_k^j) - e_k^j)^T (\hat{v}_k^j - y)}_{\Lambda}. \quad (32)$$

Typically, at this step one would use convexity of  $f_j(x)$  to simplify the term  $\Lambda$  in Eq. 32. However, since the gradient of  $f_j(x)$  is perturbed by noise  $e_k^j$ , and hence we need to follow a few more steps before we arrive at the iterate lemma.

We consider the fused state iterates  $\hat{v}_k^j$ , the average  $\bar{v}_k$  and the deviation of iterate from the average,

$$q_k^j = \hat{v}_k^j - \bar{v}_k. \quad (33)$$

We also derive a simple inequality here that will be used later,  $\|q_k^j\| \leq \max_{j \in \mathcal{V}} \|\delta_k^j\|$ .

$$\begin{aligned} \|q_k^j\| &= \|\hat{v}_k^j - \bar{v}_k\| = \left\| \sum_i B_k[j, i] x_k^i - \bar{v}_k \right\| && \text{from Eq. 12} \\ &= \left\| \sum_i B_k[j, i] x_k^i - \bar{x}_k \right\| && \bar{x}_k = \bar{v}_k \\ &\leq \sum_i B_k[j, i] \|x_k^i - \bar{x}_k\| \leq \left( \sum_i B_k[j, i] \right) \max_{i \in \mathcal{V}} \|x_k^i - \bar{x}_k\| \\ &\leq \max_{j \in \mathcal{V}} \|\delta_k^j\| && \text{from Eq. 3} \end{aligned} \quad (34)$$

Note that similarly, we can derive another inequality that will be used later,  $\sum_j \|\hat{v}_k^j - y\| \leq \sum_j \|x_k^j - y\|$ .

$$\begin{aligned} \sum_j \|\hat{v}_k^j - y\| &= \sum_j \left\| \sum_i B_k[j, i] x_k^i - y \right\| && \text{from Eq. 12} \\ &= \sum_j \left\| \sum_i B_k[j, i] (x_k^i - y) \right\| && B_k \text{ is row stochastic} \\ &\leq \sum_j \sum_i B_k[j, i] \|x_k^i - y\| = \sum_i \left( \sum_j B_k[j, i] \right) \|x_k^i - y\| \\ &\leq \sum_i \|x_k^i - y\| && B_k \text{ is column stochastic} \end{aligned} \quad (35)$$

We use gradient Lipschitzness assumption and the fact that  $\bar{x}_k = \bar{v}_k$  (doubly stochastic matrix fusion does not change the average, cf. [53]) to arrive at the following relation,

$$\nabla f_j(v_k^j) = \nabla f_j(\bar{v}_k) + l_k^j, \quad (36)$$

<sup>6</sup> An alternate way to prove this inequality would be to follow the same process used to prove Eq. 35 except that we start with squared terms and use the fact that  $\sum_j (B_k[j, i])^2 \leq 1$ .

Next, we bound the vector  $l_k^j$ ,

$$\begin{aligned}
\max_{j \in \mathcal{V}} \|l_k^j\| &\leq \max_{j \in \mathcal{V}} \{N\|v_k^j - \bar{v}_k\|\}, \\
\max_{j \in \mathcal{V}} \|l_k^j\| &\leq \max_{j \in \mathcal{V}} \{N\|\hat{v}_k^j - \bar{v}_k\| + \alpha_k N\|e_k^j\|\}, && \text{from Eq. 12 and Triangle Inequality} \\
&\leq N \left( \max_{j \in \mathcal{V}} \|q_k^j\| + \alpha_k \Delta \right) && \|e_k^j\| \leq \Delta
\end{aligned} \tag{37}$$

We use above expressions to bound the term  $\Lambda$ , in Eq. 32. We use  $\hat{v}_k^j = \bar{v}_k + q_k^j$  from Eq. 33 and the gradient relation in Eq. 36 to get,

$$\begin{aligned}
\Lambda &= 2\alpha_k \sum_{j=1}^S \left[ (\nabla f_j(v_k^j) - e_k^j)^T (y - \hat{v}_k^j) \right] \\
&= 2\alpha_k \sum_{j=1}^S \left[ (\nabla f_j(\bar{v}_k) - e_k^j + l_k^j)^T (y - \bar{v}_k - q_k^j) \right] \\
\Lambda &= 2\alpha_k [T_1 + T_2 + T_3], \text{ where,}
\end{aligned}$$

$$T_1 = \sum_{j=1}^S (\nabla f_j(\bar{v}_k) - e_k^j)^T (y - \bar{v}_k), \quad T_2 = \sum_{j=1}^S (\nabla f_j(\bar{v}_k) - e_k^j)^T (-q_k^j), \quad \text{and} \quad T_3 = \sum_{j=1}^S (l_k^j)^T (y - \hat{v}_k^j).$$

Individually  $T_1$ ,  $T_2$  and  $T_3$  can be bound as follows,

$$\begin{aligned}
T_1 &= \left( \sum_{j=1}^S \nabla f_j(\bar{v}_k) - e_k^j \right)^T (y - \bar{v}_k) = \nabla f(\bar{v}_k)^T (y - \bar{v}_k) && \sum_j e_k^j = 0 \\
&\leq f(y) - f(\bar{v}_k) && f(x) \text{ is convex}
\end{aligned} \tag{38}$$

$$\begin{aligned}
T_2 &\leq \sum_{j=1}^S \|\nabla f_j(\bar{v}_k) - e_k^j\| \|(-q_k^j)\| \\
&\leq (L + \Delta) S \max_{j \in \mathcal{V}} \|q_k^j\| \\
&\leq (L + \Delta) S \max_{j \in \mathcal{V}} \|\delta_k^j\| && \text{from Eq. 34}
\end{aligned} \tag{39}$$

$$\begin{aligned}
T_3 &= \sum_{j=1}^S (l_k^j)^T (y - \hat{v}_k^j) \\
&\leq \max_{j \in \mathcal{V}} \|l_k^j\| \sum_{j=1}^S \|\hat{v}_k^j - y\| \\
&\leq N \left( \max_{j \in \mathcal{V}} \|q_k^j\| + \alpha_k \Delta \right) \sum_{j=1}^S \|\hat{v}_k^j - y\| && \text{from Eq. 37} \\
&\leq N \left( \max_{j \in \mathcal{V}} \|\delta_k^j\| + \alpha_k \Delta \right) \sum_{j=1}^S \|\hat{v}_k^j - y\| && \text{from Eq. 34}
\end{aligned}$$

We further use  $2\|a\| \leq 1 + \|a\|^2$  to bound term  $T_3$ .

$$\begin{aligned}
T_3 &\leq N \left( \max_{j \in \mathcal{V}} \|\delta_k^j\| + \alpha_k \Delta \right) \sum_{j=1}^S \|\hat{v}_k^j - y\| \\
&\leq N \left( \max_{j \in \mathcal{V}} \|\delta_k^j\| + \alpha_k \Delta \right) \left[ \sum_{j=1}^S \|x_k^j - y\| \right] && \text{from Eq. 35} \\
&\leq \frac{N}{2} \left( \max_{j \in \mathcal{V}} \|\delta_k^j\| + \alpha_k \Delta \right) \left[ \sum_{j=1}^S \left( 1 + \|x_k^j - y\|^2 \right) \right] && 2\|a\| \leq 1 + \|a\|^2
\end{aligned} \tag{40}$$

We combine the bounds on  $T_1, T_2$  and  $T_3$  (Eq. 38, 39 and 40) to get,

$$\Lambda \leq -2\alpha_k (f(\bar{v}_k) - f(y)) + 2\alpha_k S \max_{j \in \mathcal{V}} \|\delta_k^j\| (L + \Delta) + \alpha_k N \left( \max_{j \in \mathcal{V}} \|\delta_k^j\| + \alpha_k \Delta \right) [S + \eta_k^2] \quad (41)$$

Note that we can replace,  $f(\bar{v}_k)$  with  $f(\bar{x}_k)$ . This follows from the fact that doubly stochastic matrices preserve iterate averages, i.e.  $\bar{v}_k = \bar{x}_k$ . The iterate update relation hence becomes,

$$\eta_{k+1}^2 \leq (1 + F_k) \eta_k^2 - 2\alpha_k (f(\bar{x}_k) - f(y)) + H_k, \quad (42)$$

where,  $F_k = \alpha_k N \left( \max_{j \in \mathcal{V}} \|\delta_k^j\| + \alpha_k \Delta \right)$  and  $H_k = 2\alpha_k S \max_{j \in \mathcal{V}} \|\delta_k^j\| (L + N/2 + \Delta) + \alpha_k^2 S [N\Delta + (L + \Delta)^2]$ .  $\square$

### B.3 Proof of Theorem 1 (Convergence of RSS)

*Proof.* We intend to prove convergence using Lemma 3. We begin by using the relation between iterates given in Lemma 2 with  $y = x^* \in \mathcal{X}^*$ ,

$$\eta_{k+1}^2 \leq (1 + F_k) \eta_k^2 - 2\alpha_k (f(\bar{x}_k) - f(y)) + H_k \quad (43)$$

We check if the above inequality satisfies the conditions in Lemma 3 viz.  $\sum_{k=0}^{\infty} F_k < \infty$  and  $\sum_{k=0}^{\infty} H_k < \infty$ .  $F_k$  and  $H_k$  are defined in Lemma 2.

We first show that  $\sum_{k=0}^{\infty} \alpha_k \max_{j \in \mathcal{V}} \|\delta_k^j\| < \infty$  using the expression for state disagreement from average given in Lemma 1.

$$\sum_{k=0}^{\infty} \alpha_k \max_{j \in \mathcal{V}} \|\delta_k^j\| \leq \underbrace{S\theta \max_{i \in \mathcal{V}} \|x_0^i\| \sum_{k=0}^{\infty} \alpha_k \beta^{k+1}}_{U_1} + \underbrace{S\theta(L + \Delta) \sum_{k=0}^{\infty} \alpha_k \sum_{l=1}^k \beta^{k+1-l} \alpha_{l-1}}_{U_2} + \underbrace{2(L + \Delta) \sum_{k=0}^{\infty} \alpha_k^2}_{U_3}$$

The first term  $U_1$  can be shown to be convergent by using the ratio test. We observe that,

$$\limsup_{k \rightarrow \infty} \frac{\alpha_{k+1} \beta^{k+2}}{\alpha_k \beta^{k+1}} = \limsup_{k \rightarrow \infty} \frac{\alpha_{k+1} \beta}{\alpha_k} < 1 \Rightarrow \sum_{k=0}^{\infty} \alpha_k \beta^k < \infty,$$

since,  $\alpha_{k+1} \leq \alpha_k$  and  $\beta < 1$ . Now we move on to show that  $U_2$  is finite. It follows from  $\alpha_k \leq \alpha_l$  (where  $l \leq k$ ), and Lemma 4 ( $\sum_k \alpha_k^2 < \infty$ ),

$$\sum_{k=0}^{\infty} \alpha_k \sum_{l=1}^k \beta^{k+1-l} \alpha_{l-1} \leq \sum_{k=0}^{\infty} \sum_{l=1}^k \beta^{k+1-l} \alpha_{l-1}^2 < \infty.$$

$U_3$  is finite because  $\sum_k \alpha_k^2 < \infty$  (Eq. 4). Since we have shown,  $U_1 < \infty$ ,  $U_2 < \infty$ , and  $U_3 < \infty$ , we conclude  $\sum_{k=0}^{\infty} \alpha_k \max_{j \in \mathcal{V}} \|\delta_k^j\| < \infty$ .

Clearly,  $\sum_{k=0}^{\infty} F_k < \infty$  and  $\sum_{k=0}^{\infty} H_k < \infty$ , since we proved that  $\sum_{k=0}^{\infty} \alpha_k \max_{j \in \mathcal{V}} \|\delta_k^j\| < \infty$  and we know that  $\sum_k \alpha_k^2 < \infty$ . We can now apply Lemma 3 to Eq. 43 and conclude  $\sum_{k=0}^{\infty} 2\alpha_k (f(\bar{x}_k) - f(x^*)) < \infty$ .

We use  $\sum_{k=0}^{\infty} 2\alpha_k (f(\bar{x}_k) - f(x^*)) < \infty$  to show the convergence of the iterate-average to the optimum. Since we know  $\sum_{k=0}^{\infty} \alpha_k = \infty$ , it follows directly that  $\liminf_{k \rightarrow \infty} f(\bar{x}_k) = f(x^*)$  (an alternate proof for this statement is provided later in the appendix). And due to the continuity of  $f(x)$ , we know that the sequence of iterate average must enter the optimal set  $\mathcal{X}^*$  (i.e.  $\bar{x}_k \in \mathcal{X}^*$ ). Since  $\mathcal{X}^*$  is bounded (compactness in  $\mathbb{R}^D$ ), we know that there exists a iterate-average sub-sequence  $\bar{x}_{k_l} \leq \bar{x}_k$  that converges to some  $x^* \in \mathcal{X}^*$ . This follows from the Bolzano-Weierstrass theorem.

Also note that Lemma 3 states that  $\eta_k^2$  has a finite limit. Let  $\lim_{k \rightarrow \infty} \eta_k^2 = \eta_{x^*}$  ( $\forall x^* \in \mathcal{X}^*$ ).

$$\begin{aligned}
\lim_{k \rightarrow \infty} \eta_k^2 &= \lim_{k \rightarrow \infty} \sum_{i=1}^S \|x_k^i - x^*\|^2 = \lim_{k \rightarrow \infty} \sum_{i=1}^S \|\bar{x}_k + \delta_k^i - x^*\|^2 \\
&= \lim_{k \rightarrow \infty} \sum_{i=1}^S [\|\bar{x}_k - x^*\|^2 + \|\delta_k^i\|^2 + 2(\bar{x}_k - x^*)^T \delta_k^i] \\
&= \lim_{k \rightarrow \infty} \left[ S\|\bar{x}_k - x^*\|^2 + \sum_{i=1}^S \|\delta_k^i\|^2 + 2(\bar{x}_k - x^*)^T \left( \sum_{i=1}^S \delta_k^i \right) \right] \\
&= S \lim_{k \rightarrow \infty} \|\bar{x}_k - x^*\|^2 + \lim_{k \rightarrow \infty} \sum_{i=1}^S \|\delta_k^i\|^2 & \sum_i \delta_k^i = 0 \text{ by definition (Eq. 3)} \\
&= S \lim_{k \rightarrow \infty} \|\bar{x}_k - x^*\|^2 \triangleq \eta_{x^*} & \lim_{k \rightarrow \infty} \max_{j \in \mathcal{V}} \|\delta_k^j\| = 0 \text{ from Claim 1}
\end{aligned}$$

From the statement above, we know  $\lim_{k \rightarrow \infty} \|\bar{x}_k - x^*\| = \eta_{x^*}$ . Due to  $\liminf_{k \rightarrow \infty} f(\bar{x}_k) = f(x^*)$  and continuity of  $f(x)$ , we know that the iterate average  $\bar{x}_k$  is inside the optimal set infinitely often at large  $k$ . Both these statements and convexity of  $f(x)$  give imply that the iterate average converges to a unique point  $x^* \in \mathcal{X}^*$ .

**Claim 1** (Consensus). *All agents asymptotically reach consensus, i.e.  $\lim_{k \rightarrow \infty} \|x_k^i - x_k^j\| = 0$ , for all  $i, j$ .*

We know from Claim 1 that the agents agree to a parameter vector asymptotically (i.e.  $x_k^j \rightarrow x_k^i$ ,  $\forall i \neq j$  as  $k \rightarrow \infty$ ). Hence, all agents agree to the iterate average. This along with the convergence of iterate-average to the optimal solution gives us that all agents converge to a point in optimal set  $\mathcal{X}^*$  (i.e.  $x_k^j \rightarrow x^* \in \mathcal{X}^*$ ,  $\forall j$ , as  $k \rightarrow \infty$ ). This completes the proof of Theorem 1.  $\square$

### Alternate Proof: $\liminf_{k \rightarrow \infty} f(\bar{x}_k) = f(x^*)$

*Proof.* We will prove this statement using contradiction.

We know that  $\sum_k \alpha_k (f(\bar{x}_k) - f(x^*)) < \infty$  and  $\sum_k \alpha_k = \infty$ .  $\{f(\bar{x}_k)\}$  is a sequence of real numbers and we know that  $\liminf$  always exists for this sequence (its either a real number or symbols  $\pm\infty$ ). Let us assume that  $\liminf_{k \rightarrow \infty} f(\bar{x}_k) = f(x^*) + \delta$  for some  $\delta > 0$ . Note that  $\delta$  cannot be less than 0 since  $f(x^*)$  is the minimum.

Now we know from the definition of  $\liminf$ ,  $\forall \epsilon > 0$ ,  $\exists K_0 \in \mathbb{N}$  such that  $\forall k \geq K_0$ ,

$$\begin{aligned}
\liminf_{l \rightarrow \infty} f(\bar{x}_l) - \epsilon &\leq f(\bar{x}_k), \\
f(x^*) + \delta - \epsilon &\leq f(\bar{x}_k).
\end{aligned}$$

Consider  $\epsilon = \delta/2$  and we have,  $\exists K_0$  such that,  $\forall k \geq K_0$  such that,

$$f(x^*) + \frac{\delta}{2} \leq f(\bar{x}_k) \implies f(\bar{x}_k) - f(x^*) \geq \frac{\delta}{2}.$$

Let us consider  $C \triangleq \sum_{k=0}^{\infty} \alpha_k (f(\bar{x}_k) - f(x^*)) < \infty$ .

$$\begin{aligned}
C &\triangleq \sum_{k=0}^{\infty} \alpha_k (f(\bar{x}_k) - f(x^*)) = \underbrace{\sum_{k=0}^{K_0} \alpha_k (f(\bar{x}_k) - f(x^*))}_{T1} + \underbrace{\sum_{k=K_0+1}^{\infty} \alpha_k (f(\bar{x}_k) - f(x^*))}_{T2} \\
&\geq \underbrace{\sum_{k=0}^{K_0} \alpha_k (f(\bar{x}_k) - f(x^*))}_{T1} + \underbrace{\sum_{k=K_0+1}^{\infty} \alpha_k \frac{\delta}{2}}_{T2} \tag{44}
\end{aligned}$$

$T1$  is finite since  $C$  is finite.  $T2$  grows unbounded since  $\sum_{k=K_0+1}^{\infty} \alpha_k = \infty$ . Substituting both  $T1$  and  $T2$  in Eq. 44 we get  $C \geq \infty$  in contradiction to the definition of  $C$  (being finite). Hence  $\delta = 0$ , implying  $\liminf_{k \rightarrow \infty} f(\bar{x}_k) = f(x^*)$ .  $\square$

## B.4 Proof of Claim 1 (Consensus Claim)

*Proof.* We begin with the iterate disagreement relation in Lemma 1,

$$\max_{j \in \mathcal{V}} \|\delta_{k+1}^j\| \leq \underbrace{S\theta\beta^{k+1} \max_{i \in \mathcal{V}} \|x_0^i\|}_{V_1} + \underbrace{S\theta(L + \Delta) \sum_{l=1}^k \beta^{k+1-l} \alpha_{l-1}}_{V_2} + \underbrace{2\alpha_k(L + \Delta)}_{V_3} \quad (45)$$

The first term  $V_1$  decreases exponentially with  $k$ . Hence, for any  $\epsilon > 0$ ,  $\exists K_1 (= \lceil \log_\beta \frac{\epsilon}{3S\theta \max_{i \in \mathcal{V}} \|x_0^i\|} \rceil)$  such that,  $\forall k > K_1$ , we have  $V_1 < \epsilon/3$ .

The second term is more complicated. We split it into two parts  $A$  and  $B$ , using the fact that  $\exists K$  such that,  $\alpha_k < \xi$ ,  $\forall k \geq K_2$ , due to the non-increasing property of  $\alpha_k$ .

$$\sum_{i=1}^k (\alpha_i \beta^{k-i}) = \underbrace{(\alpha_0 \beta^k + \dots + \alpha_{K_2-1} \beta^{k-K_2+1})}_A + \underbrace{(\alpha_{K_2} \beta^{k-K_2} + \dots + \alpha_k \beta^0)}_B$$

We can bound the terms  $A$  and  $B$ .

$$\begin{aligned} A &= \alpha_0 \beta^k + \alpha_1 \beta^{k-1} + \dots + \alpha_{K_2-1} \beta^{k-K_2+1} \\ &\leq \alpha_0 (\beta^k + \dots + \beta^{k-K_2+1}) && \alpha_0 \geq \alpha_i \quad \forall i \geq 1 \\ &\leq \alpha_0 \beta^{k-K_2+1} \left( \frac{1 - \beta^{K_2}}{1 - \beta} \right) \leq \frac{\alpha_0 \beta^{k-K_2+1}}{1 - \beta} && \beta < 1 \end{aligned} \quad (46)$$

$$\begin{aligned} B &= \alpha_{K_2} \beta^{k-K_2} + \dots + \alpha_k \beta^0 \\ &< \xi \beta \left( \frac{1 - \beta^{k-K_2+1}}{1 - \beta} \right) \leq \frac{\xi \beta}{1 - \beta} && \alpha_i < \xi, \quad \forall i \geq K_2 \end{aligned} \quad (47)$$

The right side of inequality in Eq. 46 is monotonically decreasing in  $k$  ( $\beta < 1$ ) with limit 0 as  $k \rightarrow \infty$ . Hence  $\exists K_3 > K$  such that  $A < \epsilon/6$ ,  $\forall k \geq K_3$ .

Since  $\alpha_k$  is non-increasing,  $\exists K_2$  such that  $\alpha_i < \epsilon(1 - \beta)/6\beta$  for all  $k \geq K_2$  (we get this by assuming  $\xi = \epsilon(1 - \beta)/6\beta$  (in Eq. 47)). This results in  $B < \epsilon/6$ , for all  $k \geq K_2$ . Hence,  $\exists K_4 = \max\{K_2, K_3\}$  such that  $(A + B) < \epsilon/3$  for all  $k > K_4$ . We can easily show that, since  $A + B$  can arbitrarily decrease with  $\epsilon$ ,  $V_2 = S\theta(L + \Delta)(A + B)$  also decreases. Hence, there exists  $K_5$  such that  $V_2 < \epsilon/3$  for all  $k \geq K_5$ .

The third term  $V_3$  decreases at the same rate as  $\alpha_k$ . Hence, for any  $\epsilon > 0$ ,  $\exists K_6$  ( $k$  such that  $\alpha_k \leq \frac{\epsilon}{6(L + \Delta)}$ ), such that  $\forall k > K_6$ , we have  $V_3 < \epsilon/3$ .

We have convergence based on  $\epsilon$ - $\delta$  definition of limits. For any  $\epsilon > 0$ , there exists  $K_{\max} = \max\{K_1, K_5, K_6\}$  such that  $\max_{j \in \mathcal{V}} \|\delta_k^j\| \leq V_1 + V_2 + V_3 < \epsilon$  for all  $k \geq K_{\max}$ . This implies that,

$$\lim_{k \rightarrow \infty} \max_{j \in \mathcal{V}} \|\delta_k^j\| = \lim_{k \rightarrow \infty} \max_{j \in \mathcal{V}} \|x_k^j - \bar{x}_k\| = 0.$$

□